Changing Viewpoint: From ReLU Units to Context-Selected Linear Mappings

Some ideas are difficult to communicate not because they require difficult mathematics, but because they require a change in the way the reader mentally represents the object being discussed.

Concatenated Context Selected Linear Mappings (CCSLM) is one such idea.

The mathematics is relatively straightforward. The conceptual shift is not. To see CCSLM clearly, it is useful to make several changes of viewpoint explicitly rather than assuming that the reader will make them automatically.

The central shift is from looking at **individual activation functions** to looking at **what a group of activation functions does to the system as a whole**.

1. From the local viewpoint to the system viewpoint

Consider an ordinary ReLU neuron:

$$y = \mathrm{ReLU}(x).$$

At the local level, we naturally describe this as an activation function. We ask whether the neuron is active, what its activation is, and how its derivative behaves.

That is a useful description, but it is a local one.

Now consider many ReLU functions operating together inside a layer. Each ReLU produces a binary decision:

$$d_i = \begin{cases} 1 & x_i > 0 \\ 0 & x_i \le 0. \end{cases}$$

Collect those decisions into a diagonal matrix:

$$D(x) = \mathrm{diag}(d_1, d_2, \ldots, d_n).$$

The entire collection of ReLU decisions can now be regarded as a **context**.

The important change is therefore:

> Instead of asking what each ReLU does individually, ask what the entire collection of ReLUs does to the computation.

The answer is that it selects a particular linear transformation.

For example,

$$y = WD(x)x.$$

For any particular activation pattern, $D(x)$ is fixed. Consequently, the network is applying one particular linear mapping.

The ReLUs have therefore done something slightly different from the way we normally describe them.

They have:

1. examined the incoming state,
2. generated a context,
3. selected a linear mapping,
4. allowed the main data to pass through that mapping.

This is the first major viewpoint shift.

---

2. From "nonlinearity" to "context plus linear computation"

The usual description of ReLU is that it introduces nonlinearity.

That is correct, but it hides a useful decomposition.

Instead of thinking of

$$y = W\,\mathrm{ReLU}(x)$$

as one indivisible nonlinear operation, we can decompose its function into two conceptual roles.

The activation functions produce **context**.

That context determines a **linear operator**.

The data then flows through that operator.

Schematically:

$$\boxed{x} \quad \longrightarrow \quad \boxed{\text{context generation}} \quad \longrightarrow \quad \boxed{\text{linear mapping}} \quad \longrightarrow \quad \boxed{y}.$$

The context-generating mechanism and the data transformation are therefore conceptually distinct even though they are physically interwoven in an ordinary neural-network implementation.

This is a decomposition of the computation rather than a claim that the original implementation contains two separate machines.

The same operation can consequently be viewed in two ways:

Local viewpoint

> ReLU neurons transform their inputs nonlinearly.

System viewpoint

> A collection of ReLU decisions establishes a context, and that context selects the linear transformation applied to the data.

Neither description is mathematically wrong. The second simply exposes a different level of organization.

---

3. From individual neurons to an operator

The next shift is from thinking about neurons as the fundamental computational objects to thinking about **linear mappings as the fundamental data transformations**.

Suppose a layer has several possible contexts:

$$C_1, C_2, \ldots, C_k.$$

Each context corresponds to a linear mapping:

$$W(C_1), W(C_2), \ldots, W(C_k).$$

The layer can then be viewed as

$$y = W(C)x.$$

The individual ReLU activations have disappeared from the description because their collective effect has been summarized by C.

This is important.

We have not thrown away the ReLU functions. We have **lifted the description to a higher level**.

The ReLUs are still doing the work. We are simply describing the result of their collective action rather than describing every local operation.

This is analogous to describing a computer program in terms of the operation it performs rather than the individual transistor states that implement it.

---

4. From one mapping to a sequence of mappings

Once a single layer is viewed this way, the same viewpoint can be applied repeatedly.

A multilayer system becomes

$$x \to W_1(C_1) \to W_2(C_2) \to W_3(C_3) \to \cdots \to y.$$

This is the central CCSLM picture.

The system is a **concatenation of context-selected linear mappings**.

The contexts do not necessarily have to be identical.

One layer might respond primarily to one aspect of state. Another might respond to something completely different.

For a robot, for example:

$$C_1 = \text{operating mode},$$

$$C_2 = \text{speed},$$

$$C_3 = \text{time or position}.$$

The result is still one sequential computational system:

$$W_3(C_3)W_2(C_2)W_1(C_1)x.$$

There is no requirement for a large collection of separately executing pathways. The system remains a chain of transformations.

Context simply determines which transformation is used at each stage.

---

5. From "the network" to "data channel plus control channel"

Another useful viewpoint shift is to distinguish **what is being transformed** from **what controls the transformation**.

The main data channel carries the quantities being processed:

$$x \to W(C)x.$$

The context acts through a separate conceptual control channel:

$$C \to W(C).$$

Thus:

$$\boxed{\text{context/control}} \quad \longrightarrow \quad \boxed{\text{choice of transformation}}$$

while simultaneously:

$$\boxed{\text{data}} \quad \longrightarrow \quad \boxed{\text{selected transformation}} \quad \longrightarrow \quad \boxed{\text{output}}.$$

This separation is especially useful when context does not simply mean "some additional input feature."

Context can represent persistent state, operating mode, time, position, speed, phase, goal, or any other variable that should influence **how the computation is performed**.

The distinction is therefore:

> Data says *what is being processed*.
> Context says *how it should be processed*.

---

6. The shift is a generalization upward

These viewpoint changes can be summarized as a movement upward through levels of description:

$$\text{ReLU} \rightarrow \text{activation pattern} \rightarrow \text{selected linear mapping} \rightarrow \text{sequence of selected mappings} \rightarrow \text{context-controlled}$$

At each stage, some lower-level detail is compressed into a higher-level object.

The ReLUs become an activation pattern.

The activation pattern becomes a context.

The context identifies a linear mapping.

A sequence of such mappings becomes a computational system.

This is why CCSLM can initially feel strangely abstract. It is not asking the reader to learn an entirely new mathematical operation. It is asking the reader to **change the level at which the existing operation is being observed**.

The transformation is therefore partly one of abstraction.

We move from:

> "What is this neuron doing?"

to:

> "What is this layer doing?"

and then to:

> "What computational state is this layer selecting?"

and finally:

> "What kind of system results when these context-selected transformations are concatenated?"

---

7. ReLU as both activation function and context generator

This viewpoint also changes the interpretation of ReLU itself.

A ReLU can be regarded as having two consequences.

First, it contributes to the **context** by determining whether a particular component is active.

Second, through that collective context, the ReLU population determines which **linear mapping** the data encounters.

Thus the ReLU network can be conceptually decomposed into:

$$\boxed{\text{ReLU population}} \rightarrow \boxed{\text{context}} \rightarrow \boxed{\text{linear mapping}}.$$

The familiar nonlinear behaviour has not disappeared. It has been relocated.

The nonlinearity arises because the mapping changes when the context changes.

Within a particular context, however, the resulting computation is linear.

This gives a system-level interpretation of the familiar piecewise-linear nature of ReLU networks:

> **The network moves between different linear mappings as its context changes.**

That is a very different mental picture from seeing nonlinearity as something that every individual neuron continuously contributes to the computation.

---

8. Why the viewpoint matters

Once this viewpoint has been adopted, questions that are difficult to see in the conventional neuron-centric description become natural.

For example:

- How different are the linear mappings associated with different contexts?
- Do the mappings form clusters?
- Which sources of context produce meaningful specialization?
- Can context be supplied independently of the data?
- Can different layers respond to different context variables?
- How much computational specialization can be obtained without creating explicit branches?
- What happens when temporal or persistent state is used as context?
- Can a system deliberately construct useful contexts rather than merely discovering them incidentally?

These are system-level questions.

They are difficult to formulate if the fundamental object of thought is the individual activation function.

CCSLM therefore represents a change in emphasis:

from neurons to mappings, from activations to contexts, from local operations to system behaviour, and from a single network computation to a sequence of context-selected computational states.

The purpose of the viewpoint is not to deny the conventional description of neural networks.

It is to add another level of description above it.

And that higher-level description may make some properties of neural computation easier to see, analyse, design, and generalize.